AFRL-RI-RS-TR-2010-035
Final Technical Report
January 2010

FIRST TIME AUTHENTICATION FOR AIRBORNE NETWORKS (FAAN)

Distributed Infinity

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

STINFO COPY

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report was cleared for public release by the 88th ABW, Wright-Patterson AFB Public Affairs Office and is available to the general public, including foreign nationals. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RI-RS-TR-2010-035 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE DIRECTOR:

/s/
BRADLY S. PAUL, 2Lt, USAF
Work Unit Manager

/s/
WARREN H. DEBANY, Jr.
Technical Advisor, Information Grid Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
JANUARY 2010	Final	July 2008 – October 2009

4. TITLE AND SUBTITLE

FIRST TIME AUTHENTICATION FOR AIRBORNE NETWORKS (FAAN)

5a. CONTRACT NUMBER
FA8750-08-C-0228

5b. GRANT NUMBER
N/A

5c. PROGRAM ELEMENT NUMBER
33140F

6. AUTHOR(S)

Wendy L. Hamilton

5d. PROJECT NUMBER
7820

5e. TASK NUMBER
PM

5f. WORK UNIT NUMBER
01

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Distributed Infinity
1382 Quartz Mountain Drive
Larkspur, CO 80118-8217

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

AFRL/RIGH
525 Brooks Road
Rome NY 13441-4505

10. SPONSOR/MONITOR'S ACRONYM(S)
N/A

11. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFRL-RI-RS-TR-2010-035

12. DISTRIBUTION AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED. PA# 88ABW-2010-0302 Date Cleared: 25-January-2010

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Mobile airborne networks deployed and used by the armed forces face significant challenges in balancing security concerns with reliably servicing the needs of the forces dependent on it. These issues are complicated by the ever-changing collaborative environments that these networks are tasked to support. The Zero Knowledge Protocol (ZKP) is a powerful technique that replaces the portion of a traditional credential-based key management system where credentials are exchanged. Instead, in ZKPs a proof of the existence of the credential is sent, allowing the receiver to develop absolute confidence that the user has a valid credential, without exposing the credential to risk of compromise. Based on the results of this research, we assert that it may be possible to construct a ZKP scheme with strong anonymity given any identification protocol and a shared-key update scheme.

15. SUBJECT TERMS

Zero Knowledge Protocol, Anonymity, Identification, Authentication, Revocation

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Bradly S. Paul,2Lt, USAF
U	U	U	UU	40	19b. TELEPHONE NUMBER (*Include area code*) N/A

Table of Contents

List of Figures

List of Tables

1. Overview of First Time Authentication for Airborne Networks

Mobile airborne networks deployed and used by the armed forces face significant challenges in balancing security concerns with reliably servicing the needs of the forces dependent on it. These issues are complicated by the ever-changing collaborative environments that these networks are tasked to support. In this program, we address one of the most significant challenges inherent in these networks: how do you enable valid new users or hardware to securely interact with a network at the appropriate access level, where the network and user are both unknown to each other?

There are several major hurdles to overcome when tackling this issue. First, shifts in the makeup of allied forces over time may mean that large numbers of known users may lose access or new users gain access privilege in short periods of time. Second, different portions of different networks are often operated by separate factions, and users are appropriately reticent to send credential information through some of these networks. This is particularly true in MANETs (Mobile Adhoc Network) that may be self-organizing, because of the risk that this credential information may become compromised, and inappropriately utilized at a future date through replay or man-in-the-middle attacks. For example, consider a military operation scenario where a soldier with time critical targeting information must quickly connect to an Adhoc coalition network to impart necessary information, but has never accessed that network before. The information must be quickly and reliably passed, while maintaining anonymity and security from unknown entities that may be monitoring transmissions.

Essentially, the fundamental issues from both the user and network perspective boil down to the same thing: credential and key management systems. Whether they are meant to assure networks of a user's privileges or a user of the network's validity, current credential and key management systems are designed for long-term use on the order of hours or days. In a collaborative airborne networking environment the fundamental nature of the mission encourages frequent user and hardware turnover on the order of minutes or seconds.

In the First Time Authentication for Airborne Networks (FAAN) program we investigate the use of the Zero Knowledge Protocol (ZKP) to authenticate and secure communication in a manner compatible with the requirements of an Airborne network.

2. Zero Knowledge Protocol Algorithms

The Zero Knowledge Protocol (ZKP) is a powerful technique that replaces the portion of a traditional credential-based key management system where credentials are exchanged. Instead, in ZKPs a proof of the existence of the credential is sent, allowing the receiver to develop absolute confidence that the user has a valid credential, without exposing the credential to risk of compromise. The proof consists of a series of challenge-response exchanges, and even the capture of all previous challenge-response interchanges does not allow the re-creation of the credential, in the same way that the capture of encrypted traffic does not allow the cracking of an encryption key. In addition, the mathematics of the protocol guarantees that no information other than the existence of a valid certificate is contained in the response. Thus, unlike with a standard scheme where repeat users can be identified or tracked using their unique certificate, such information can only be gleaned from the exchange if explicitly included by the user. This property can be valuable for certain personnel for which identity, time, location, and content of transmittals are extremely sensitive and leakage of such information could pose serious risk to themselves and their mission.

In an open environment, the exchanges of authentication protocols are susceptible to eavesdropping; it is important to formalize notions of security to gauge the strength of these protocols. ZKP is no exception. Methods such as Direct Anonymous Attestation [34, 35] provide properties such as anonymity, secrecy, strong secrecy, and authenticity that are desirable for any protocols to be secure; these properties are embedded in ZKP by design.

Anonymity and secrecy go hand in hand in ZKP; briefly, anonymity is when the server cannot tell who it is authenticating, and secrecy is when the user does not reveal his secret credential to the server. All users, including the adversary, may start the protocol with the server; anonymity is guaranteed because an adversary cannot tell from the protocol outputs which users are authenticated by the server. Also, an authenticated user never reveals his secret (therefore his identity) during ZKP exchanges, the server is able to establish the legitimacy of the user, but cannot tell the legitimate users apart. Furthermore, since the outputs from the user contain only the derived information of his secret key, his secrecy remains hidden.

Strong secrecy is when the adversary cannot tell when the value of the secret changes. Since the user never reveals his secret in any of its exchanges with the server, the adversary cannot tell from the outputs alone that the user has changed his secret. Authenticity is when the server is able to verify with some certainty that the user has the appropriate credentials; as we will illustrate in the following sections, when a user passes all the server's challenges, the probability that the user is an imposter becomes exponentially small, therefore authenticity is maintained in the ZKP.

2.1 ZKP Overview

In most authentication schemes, the identities of the users who wish to gain access to resources are associated with a secret identification, such as a password or certificate, known only by the users and the authority. The traditional approach is for the user to reveal their secret to the authority to prove their identity. There are several problems with this approach. First, the authority may not be who he claims to be, that is, a malicious third party could pose as the authority. In this case by revealing his secret password to the imposter the user compromises the system of trust, and the system can no longer guarantee that those who are able to gain access are legitimate users. Furthermore, even if the authority is trustworthy, revealing a password is susceptible to eavesdropping, compromising the system.

The use of zero-knowledge protocols, on the other hand, effectively removes these problems. It is usually assumed that in any zero-knowledge protocol the users have in their possession secrets that are known only to themselves. Furthermore, the secrets will generally be very hard to deduce via computation and therefore will not be susceptible to guessing by a malicious third party. Also, zero-knowledge protocols are designed such that no information is leaked about the secrets that the users possess during the authentication process, so that a malicious user would gain nothing out of eavesdropping.

A subtlety exists when a zero-knowledge protocol is being implemented. In particular, one must make a distinction between a scheme that maintains identification and anonymity. An identification scheme associates each user with a secret key, which is kept private; the server keeps the corresponding public keys, which are used upon authentication requests to validate the legitimacy of the users. However, the identity of the user is revealed once the authentication succeeds. Since the user must include his public key in his correspondence with the server, if the public/private keys are distinct then the user's activities can be followed by a malicious entity. A scheme that guarantees anonymity, on the other hand, reveals nothing about the user, even after successful authentication. A scheme with *weak anonymity* is one that assigns users with the same identification and secret information, so that all valid users look alike to any observer of the system. A scheme with *strong anonymity* assigns every user a unique identifier, yet the users remain indistinguishable to eavesdroppers and other users in the system. The generic ZKP only provides weak anonymity. We will discuss in section 4 methods that can be used to overcome this, as it inherently prevents the simultaneous support of anonymity and privilege revocation.

The general flow of zero knowledge protocols is described below. The user who wishes to prove his identity to the authority is known as the prover, and the authority is known as the verifier. The prover generally has a secret that is generated by combining a unique piece of information originally assigned when access is first granted, with a public piece of information that may be periodically published. At the beginning of each zero knowledge proof round, the prover publishes his public keys. The typical pattern of exchanges between the prover and the verifier, repeated over several rounds, is:

1) The prover (P) sends the verifier a value that is computed based on his private key.
2) The verifier flips a coin, and asks the prover to answer one of two questions based on the result of the coin flip. Generally the questions are either to answer a question about the value that was sent in step 1, or to answer a question about the secret key.
3) The prover sends the verifier the answer to his question.
4) The verifier checks that the answer is correct.

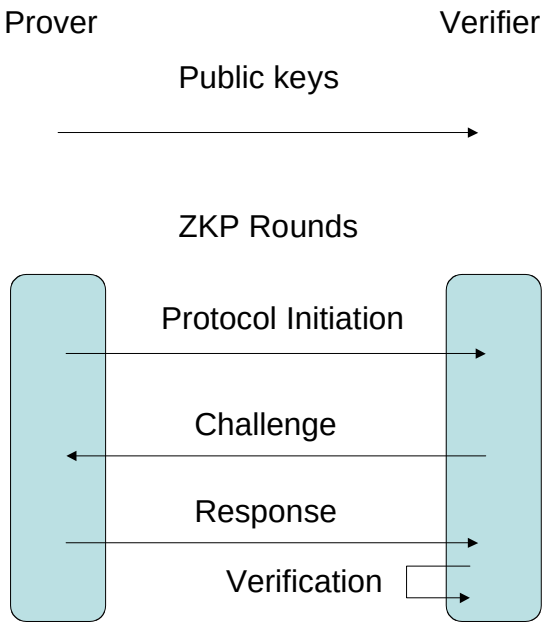

Figure 1: *Typical ZKP Exchanges*

Note that in the scenario the prover's private key is never revealed. The verifier gains more confidence in the user's authenticity as the number of rounds increase. In particular, suppose that a malicious prover *M* tries to masquerade as a legitimate user without knowing that user's private key.

Since the verifier V has no knowledge of the challenge question he's going to send until the coin flip in step 2, then since *M* doesn't have the prover's secret key M only has a 50% chance of getting the answer correct in step 3 assuming that the secret is sufficiently difficult to guess. This is a good assumption in the initial challenge-response exchange, but if the malicious prover M has been eavesdropping on a number of interchanges between legitimate users, that assumption may become invalid. A good ZKP protocol minimizes the risk of this by making intelligent secret guessing extremely difficult even if the malicious users collects significant data on valid exchanges in the same way encryption makes key guessing extremely difficult despite numerous exchange captures.

Assuming the malicious prover is unable to make an intelligent guess regarding a valid user's secret, over a *k*-round exchange between the prover and the verifier, the probability that *M* successfully cheats on all *k* rounds is $(½)^k$. For even a moderate value of *k* the probability to cheat successfully is exceptionally small.

Although all zero-knowledge protocols share the characteristics outlined above, each protocol can be categorized according to the problems that are considered computationally hard to solve. Each category of protocols comes with different intrinsic properties. In particular, we are interested in the scalability in terms of bandwidth utilization, number of exchanges, and calculation requirements for these protocols while maintaining strong security properties in an airborne networking environment.

2.2 General survey of ZKP Algorithms

The Zero Knowledge Protocol uses a computationally hard problem as the basis of its security so that without the user's secret it becomes difficult for an adversary to calculate a probable answer to any given question posed by the server during the challenge-response exchange, even if the adversary has been eavesdropping on all previous exchanges. In the past 20 years many different hard problems have been utilized. One common category of problems used are NP-Complete problems, since their computational time complexity is known to be extreme enough to foil any adversary for some subset of the possible questions (the worst case scenarios that define the time complexity). The conclusion that any NP problem can be the basis for a zero-knowledge proof is established in [2]. The Graph Three-Colorability problem is one such NP-Complete problem that has been explored for use in zero-knowledge proofs [1].

There are also non-NP problems that may be hard enough for the average case to be considered for use in ZKP. In some cases, they may even be superior. The key issue is that while NP is a description of the difficulty of answering the hardest question, it is not a description of the difficulty of answering the average question. For many NP problems there exists heuristics to help generate answers in polynomial time for a large subset of the possible questions. The utilization of such NP problems is not out of the question, but to do so a mechanism must be designed to generate a subset of the NP problem for which the average case has a high time complexity due to the ineffectiveness of heuristic solutions. Such mechanisms are not always easy, and in some cases problems that have not proven to be in either NP or P may have better average case problem characteristics making them more appropriate for use in a zero-knowledge protocol.

In [3] it is noted that when the zero knowledge requirement is relaxed to Statistical Zero Knowledge (instead of Perfect Zero Knowledge), a whole class of problems not in NP can be used to construct ZKP. These include Graph Non Isomorphism, and Permutation Group Non Isomorphism. Other problems include Quadratic Residuosity, Discrete Logarithm, as well as the Shortest Vector and Closest Vector problems in lattices.

Lattice-based Zero-Knowledge Proofs have been a popular research topic in the last decade. Two example problems that are lattice based are the Shortest Vector Problem (SVP) and Closest Vector Problem (CVP). Lattice problems rely on the hardness of finding approximate solutions to the given problem. Goldreich et. al. [4] presented some results for lattice-based ZKP and concluded that approximating SVP and CVP to within $\sqrt{n}$ (where n is the dimension of the lattice) is unlikely to be NP-hard. Micciancio [5] showed that SVP is NP-hard to approximate within any constant less than $\sqrt{2}$.

ZKP can be extended to concurrent executions of the protocol with a single prover and multiple verifiers; this concept is studied in [6]. Micciancio et. al. [7] showed some results on concurrent ZKP. An application of concurrent ZKP can be found in how a web server communicates with its clients. In this example, the web server is the prover, and the clients are the verifiers. Concurrent ZKP is designed to thwart the adversary who may control several clients and start sessions with the server in an attempt to extract information from the server.

There are also several secure identification schemes that make use of the zero knowledge protocol. The first was the Fiat-Shamir [20] scheme based on the problem of modular square roots extraction. Extensions of this scheme have been proposed, as well as some based on factorization and on the discrete log problem. These schemes have a high computational load due to arithmetic operations modulo large primes.

There are a number of schemes that have been proposed to facilitate secure identification on smart cards. In 1989 Shamir proposed a scheme based on the NP-Complete Permuted Kernels Problem (PKP) [10]. In 1992 Baritaud et.al. [16] proposed a time-memory trade-off algorithm which lead to a reduction in the computation time needed to solve PKP. In 2001 a new attack on PKP was found [17] based on an algorithm designed to count points on elliptic curves. This algorithm was still exponential, but would theoretically solve the PKP for the original size proposed by Shamir on a single PC in 125 years.

Other schemes suitable for use on smart cards have been developed [9,11,12,13,14]. In 1989 Stern [14] proposed a scheme that relied on the intractability of some coding problems, which he later claimed was not practical. In 1993 Stern proposed another scheme based on the NP-Complete problem of Syndrome Decoding [11] (SD), i.e., on the hardness of decoding a word of a given syndrome w.r.t. some binary linear error correcting code. In 1994 Stern proposed a scheme based on the NP-Complete problem of Constrained Linear Equations (CLE) [12]. In 1994 Pointcheval proposed a scheme based on the NP-Complete Perceptron Problem (PP) [13, 19]. In 1997 Poupard [15] analyzed PKP and CLE to determine the theoretical limit to the efficiency of the known attacks, to determine the practical results they permit, and to get precise evaluation of which parameters should be chosen for a secure use of these protocols. In 1999 Knudsen and Meier [18] found an attack on PP (and subsequently PPP). In 2003 Pointcheval and Poupard developed a scheme based on the NP-Complete Permuted Perceptrons Problem (PPP) [8], addressing the attacks found to date. PPP was implemented for smart cards and was able to achieve the minimal requirements of 2 KB of EEPROM, 100 Bytes of RAM, and 6.4 KB of communication.

2.3 Specific ZKP Algorithms

Below we discuss specific details of a few of the best known ZKP protocols. The Graph Isomorphism protocol was chosen at the request of AFRL to complement work being simultaneously pursued by Professor Namaduri and his students at the University of North Texas. We chose to complement this with a study of the discrete log and factoring protocols as they are the most highly studied and most rigorously reviewed problems in the community and form the basis of many cryptographic protocols, though the factoring algorithm presented here is an identification scheme which only offers weak anonymity if the public/private information is shared among users.

2.3.1 Graph Isomorphism Overview

It is generally assumed that computing the graph isomorphism between two graphs is a difficult problem [33]. However, nothing is known about the graph isomorphism (GI) class, i.e. it is unknown if GI resides in P or NP. Furthermore, there are many instances of graphs, such as planar graphs and trees, where it is known that the graph isomorphism problem can be solved efficiently in polynomial time [28]. These cases are common enough that the general form of the graph isomorphism problem is inappropriate as the basis for a zero-knowledge algorithm, since the average case analysis is poor. Thus, the question arises, is there a subset of questions which we can identify and generate that are sufficiently difficult?

One subset problem we explored is the graph isomorphic problem applied to regular graphs. It is not known if there exists an efficient isomorphism algorithm. In fact, it has been conjectured that many classes of graphs are easy to solve using heuristic based graph isomorphism analysis methods. One class of problems that are considered difficult to solve using heuristic methods are regular graphs. In fact, it has been hypothesized that regular graphs are the reason the graph isomorphism problem is not in P [31].

Under the assumption that graph isomorphism on regular graphs is a hard problem, then given two *n*-node graphs *G* and *F*, the prover(P) claims to know the mapping σ between *G* and *F*. V is the verifier.

The protocol:

1) P -> V: P generates an isomorphism *F'* of *F* using the mapping τ and sends F' to V;
2) V -> P: V sends a random bit *b*;
3) P -> V: if *b* is 0, P sends V τ; otherwise, P sends V $\sigma\,\tau$;
4) V verifies that either τ maps *F* to *F'*, or $\sigma\,\tau$ maps *G* to *F'*.
5)

2.3.2 Factoring Overview

Factoring is a problem that has been vigorously pursued in mathematics for hundreds of years, and while no proof exists experts agree it is extremely unlikely to be solved in polynomial time. Thus, while there is no proof that it is in NP, it is often treated as such. In general, one can assume that having primes *p* and *q* large enough, then it is not possible to factor $n = pq$ efficiently without prior knowledge of *p* and *q* [37]. The prover P chooses $x < n$ as its private key, and publishes *n* along with its public key *a* where $a \equiv x^2 (\mathrm{mod}\, n)$. Note that in this case anonymity is not preserved. In order to preserve anonymity all users must have the same public key.

The protocol:

1) P -> V: P chooses *r* at random, and sends $y \equiv r^2 (\mathrm{mod}\, n)$ to V;
2) V -> P: V sends a random bit *b*.
3) P -> V: if *b* is 0, then P sends V $z = r$; otherwise, P sends V $z \equiv xr (\mathrm{mod}\, n)$;
4) V verifies that $ya^b - z^2 \equiv 0 (\mathrm{mod}\, n)$.

The Guillou-Quisquater Identification Scheme [42] is an algorithm that utilizes the hardness of factoring to ensure security of the protocol.

2.3.3 Discrete Log Overview

The discrete logarithm problem [21] is simple to state: suppose that $\alpha^x \equiv \beta (\mathrm{mod}\, N)$, and that one wishes to compute the solution *x*, given α, β, and *N*. As with factoring, it has not been formally proven to be in either P or NP, but the consensus among mathematicians is that it is extremely likely to be NP. In general, with cryptographic protocols, and in particular with zero-knowledge proofs, the difficulty of the problem is used to form the basis of the security assumption, i.e., if one does not possess a solution, it is highly unlikely that one can make an intelligent guess by computing values associated with the solution in a dynamic fashion.

In the protocol, the prover P has to show his knowledge of the discrete logarithm of $\alpha^x \equiv \beta (\mathrm{mod}\, N)$ without revealing his solution. The values α, β, and *N* are published, and let *N* be a prime and Z_N^* be the integers less than *N*.

The protocol:

1) P -> V: P selects *r* from Z_N^*, and sends $\gamma \equiv \alpha^r (\mathrm{mod}\, N)$ to V;
2) V -> P: V sends a random bit *b*;
3) P -> V: P sends $y \equiv r + bx (\mathrm{mod}\, \phi(N))$ to V;
4) V checks that $\alpha^y \equiv \gamma \beta^b (\mathrm{mod}\, N)$.

The Schnorr Identification Scheme [42] is a well-known application of the discrete log protocol. Distributed Infinity has done work in the past on a more sophisticated application [39, 40, 41] of the discrete log protocol which facilitates strong anonymity through the addition of secret unique keys. These unique keys only affect updating and revocation. The general communication procedure is exactly as described here. In order to gather the results shown in section 3.2.2, it should be noted that our implementation of the Discrete log ZKP had strong anonymity support removed. This was to facilitate a head-to-head comparison of the three methods tested.

2.4 Direct Anonymous Attestation

The Direct Anonymous Attestation (DAA) protocol was proposed by the Trusted Computing Group (TCG) for use in their Trusted Platform Module (TPM) specification.

The DAA uses a discrete logarithm based ZKP as follows [34, 35]: A TPM chooses secret message f, and obtains a signature on it from the issuer via a secure two-party protocol, and then the TPM can convince a verifier that it got an attestation anonymously by showing a proof of knowledge of an attestation. This is done by computing $N = g^f$, where g is the generator of an algebraic group. The group is generally chosen such that computing the discrete logarithm is infeasible.

The TCG proposed use of the DAA requires the TPM to interact with a Privacy Certification Authority (Privacy CA) during every transaction. This bottleneck may prove to be too costly in the long run.

3. Algorithmic Requirements and Results

The theoretical requirements for a functional Zero Knowledge Protocol are detailed below as well as results obtained from our implementations of a factoring based ZKP and a discrete log based ZKP. In some cases the results are presented in term of mathematical analysis of the properties of the algorithm, such as for Graph Isomorphism. In other cases, such as with factoring and discrete log ZKPs we present performance metrics gathered by implementing specific ZKP algorithms.

3.1 Graph Isomorphism Requirements and Results

3.1.1 Requirements

3.1.1.1 Key Generation Requirements

Before the ZKP rounds begin, a candidate graph G is selected, on which the entire protocol relies. Assuming that G is chosen at random from the entire space of graphs, and that edges of a graph can be encoded using O(c) bits, then the O(n^2) bits of G and F are made public. The private key σ takes O(n) to compute, and using σ, F takes O(n^2) to generate.

However, it is naïve to assume that any random graph G is as secure as others. In particular, there are algorithms, such as Nauty[28], which can solve many graph isomorphism problems very efficiently; therefore it is vital to choose a graph G for which the isomorphism problem is hard to solve in polynomial time.

There is experimental evidence that regular graphs are harder to solve for the current state of the art algorithms [24]. Briefly, a regular graph of r-regularity is an undirected graph in which every node is adjacent to r nodes. The selection of G (and therefore, F) from an r-regular graph space as public keys would use $O(rn)$ bits as communications overhead.

Furthermore, the choice of σ is also vital to the security of the protocol. Given a graph G, the graphs induced by $Aut(G)$, or the automorphism group of G, have the same adjacency matrix structure as G. In particular, when $\sigma \in Aut(G)$, then the adjacency matrix of F looks exactly the same as G, rendering the protocol extremely easy to cheat.

Thus the choice of σ must be outside of *Aut*(*G*). Let $S_G = |Aut(G)|$ denote the size of *Aut*(*G*). The probability of choosing a non-automorphic σ is exactly $1 - (S_G/n!)$, and therefore, the expected number of picks until a non-automorphism is chosen is exactly $n!/(n! - (S_G))$. Clearly the expected number of selections for σ depends on S_G; however the exact size of *Aut*(*G*) for regular graphs is a poorly understood area [30]. Therefore we shall denote the computational overhead for selecting σ by $O(1/S_G)$, since the time it takes to find a good σ is clearly inversely proportional to S_G.

3.1.1.2 Protocol Initiation Requirements

The same issues exist here as in the key generation. Since *F* is an *r*-regular graph, then *F'* must also be an *r*-regular graph. Thus the communication overhead to send the verifier the edges of *F'* takes $O(rn)$ bits, and the computation overhead for selecting τ takes $O(1/S_F)$ where $S_F = |Aut(F)|$.

3.1.1.3 Answer Generation Requirements

Regardless of the question asked by the verifier, the prover sends one permutation to the verifier, which takes $O(n)$ bits of overhead. If the verifier asks to see the mapping between *G* and *F'*, then the prover has to compute $\sigma\tau$, which takes $O(n)$ time.

3.1.1.4 Verification Requirements

The verifier takes the permutation sent by the prover and applies it to either *G* or *F*, which takes $O(n^2)$ time to reshuffle the adjacency matrices.

3.1.1.5 Overall Performance Requirements

When regular graphs are used for the protocol, the total communications overhead is $O(rn)$ bits, and the computation overhead is $O(n^2)+O(1/S_G)+O(1/S_F)$. Clearly, the scalability of the graph isomorphism protocol relies on *r*, S_G, and S_F.

3.1.2 Theoretical Results

According to the GI protocol, the mapping σ stays fixed but a new mapping τ is generated at every iteration. It is meaningless to discuss the security of the protocol when $\sigma \in Aut(G)$, since the prover's secret is out in the open when that is the case. Therefore in the ensuing discussion we shall assume that $\sigma \notin Aut(G)$. In the following analysis, we assume an honest prover P, an honest verifier V, and an eavesdropper E who listens to the exchanges in the protocol.

***1*-round analysis**:
At step 1 of the protocol, the prover chooses τ at random to generate *F'*. The probability that $\tau \in Aut(F)$ is $S_F/n!$, and we denote the probability by Pr[$\tau \in Aut(F)$]. At step 2 & 3 of the protocol, the prover sends either the mapping τ or $\sigma\,\tau$ to the verifier, depending on the verifier's coin flip, which is random and independent of all other events in the protocol. Therefore, an eavesdropper is successful at uncovering σ only when $\tau \in Aut(F)$ and the coin flip is 1; in the language of probability and events, the Eavesdropper is successful when the conjunction of two events occurs, i.e., when $\tau \in Aut(F)$ and CoinFlip=1. The two events are independent because coin tosses are independent of everything else in the protocol, therefore the probability of this conjunction is given by

$$\Pr[\text{ CoinFlip}=1 \cap \tau \in Aut(F)] = \Pr[\text{ CoinFlip}=1] \cdot \Pr[\tau \in Aut(F)] = (1/2)(S_F/n!).$$

This analysis assumes that the eavesdropper can gain no advantage in guessing based on an analysis of previous exchanges. There is currently no theoretical basis for this assumption, which puts the validity of using GI based zero knowledge protocols into question.

***K*-round Analysis**:
We wish to find the probability that the eavesdropper successfully uncovers σ at the *k*th round. Clearly, that means the eavesdropper has failed at the first *k-1* rounds, so

$$\Pr[\text{ eavesdropper uncovers } \sigma \text{ at exactly the } k\text{th round }]$$
$$= \Big(\prod_{i=1}^{k-1} \Pr[\text{ eavesdropper fails at round } i \text{ }] \Big) \cdot \Pr[\text{ eavesdropper successful at round } k \text{ }].$$
$$= (1 - (1/2)(S_F/n!))^{k-1} \cdot (1/2)(S_F /n!).$$

Distribution of Eavesdropper's successes:
Taking the results from the *k*-round analysis, we can then calculate the probability that the Eavesdropper has to try more than *k*-rounds before he succeeds:

Pr[eavesdropper uncovers σ after the *k*th round]

= 1 - $\sum_{i=1}^{k}$ Pr[eavesdropper uncovers σ at exactly the *i*th round],

however, the quantity ($\sum_{i=1}^{k}$ Pr[eavesdropper uncovers σ at exactly the *i*th round]) is clearly a geometric series, therefore a finite sum exists. After applying the geometric sum and some simple algebraic manipulations, we arrive at

Pr[eavesdropper uncovers σ after the *k*th round] = $1 - (1 - (1/2)(S_F /n!))^k$.

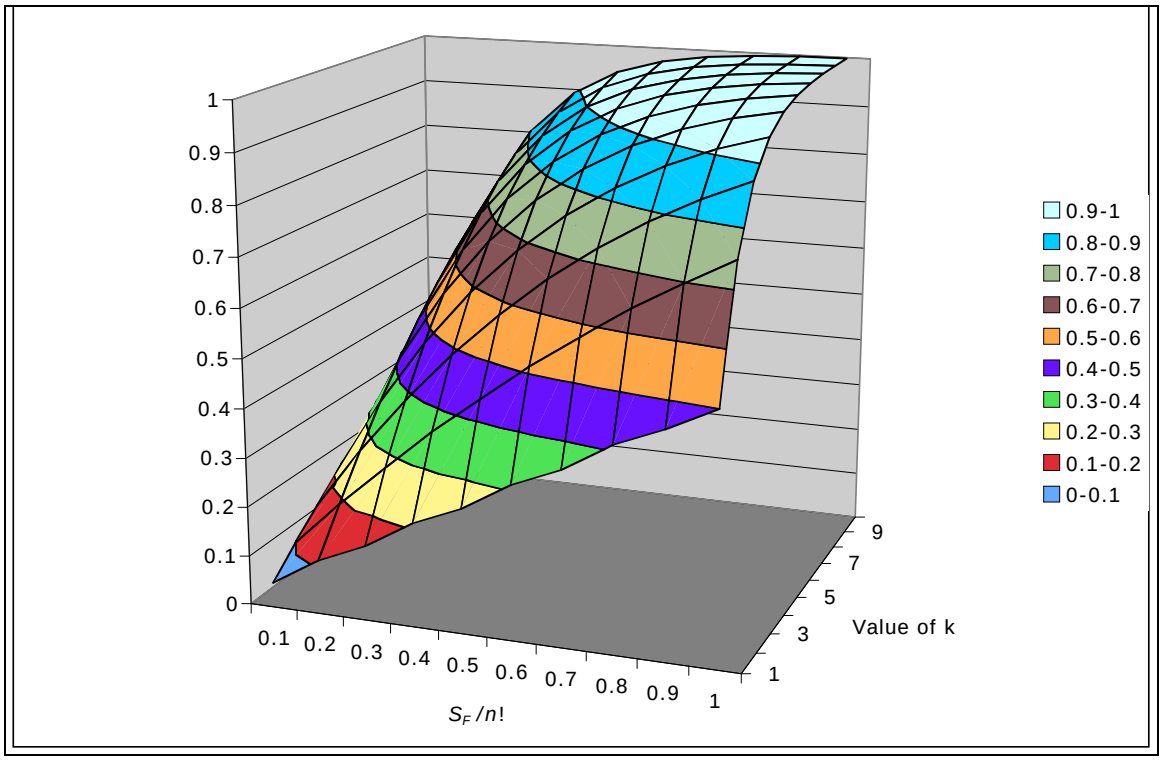

Figure 2: *Probability Distribution for uncovering secret permutation after k rounds.*

It is clear from this graph that the smaller the value of $S_F = |Aut(F)|$, the longer it takes for an eavesdropper to uncover the secret permutation σ. Therefore it can be argued that smaller values of S_F leads to a more secure GI protocol.

3.1.3 Application of the Protocol based on regular graphs.

The analysis regarding the complexity of generating graphs from the previous sections assumes that generating an appropriate graph is as easy as selecting a random graph from the space of all graphs. As is seen from the analysis, graphs whose automorphism groups are smaller tend to have better security in the protocol. Random regular graphs are not trivial to generate, however, and have been the subject of interest in the community [25, 27]. In addition, regular graphs seem to be one of the more difficult instances of graphs for graph isomorphism [31]. In this section, we examine the security of regular graphs using the technique we developed in the previous sections.

Definition: a graph G is *regular* if all vertices v in G have the same number of edges. G is *r-regular* if all vertices v in G have exactly r edges.

Bollobás [30] gave a bound on the size of automorphism groups for a regular graph G. Specifically, let θ be any permutation on the vertices of an n-node r-regular graph G; then the probability that G is invariant under θ is bounded by

$$\Pr(\theta \in Aut(G)) \leq (6rs)^{12} n^{-s(1+\varepsilon)} t^t u^u, \quad (1)$$

where s is the number of vertices moved by θ, t is the number of 2-cycles in θ, and u is the number of 3-cycles in θ. Thus, by trivial extension, the size of the automorphism group $Aut(G)$ is bounded by

$$|Aut(G)| \leq (n!) \cdot ((6rs)^{12} n^{-s(1+\varepsilon)} t^t u^u). \quad (2)$$

This bound may look formidable, but we can still draw some preliminary conclusions from it. Specifically, as the regularity of G grows, the bound suggests that the size of the automorphism group of G grows as well.

Krasikov et al. [29] considered connected regular graphs, and gave a bound that shows the structure of automorphism groups of G with respect to regularity:

$$|Aut(G)| \leq n(r!)(r-1)^{n-r-1}, \quad (3)$$

where r is the regularity of G. When $n\text{-}r \approx 0$, $|Aut(G)|$ is bounded approximately by $n!$, and when r is small, then $|Aut(G)|$ is still bounded by $n(r!)(r-1)^{n-r-1}$; in other words, when the regularity of G is close to the number of nodes of G, there exist very few distinct automorphism groups; when regularity is small compared to the number of nodes, then there exist more distinct automorphism groups.

Our conclusion based on this is that good graph selection entails the generation of a regular graph with a small automorphism group. This task does not seem at all trivial to us, but is deemed critical for successful development of this problem type into a successful zero knowledge protocol.

Application of Regular Graphs to *k*-round GI Protocol:

One often employs the use of confidence level, *c*, to measure the robustness of protocols. Taking into account the results we derived in the previous sections, a useful metric to measure the strength of the GI protocol is

$$\max_k \Pr[\text{ eavesdropper uncovers } \sigma \text{ after } (k+1)\text{th round }] \geq 1 - c;$$

that is, we wish to maximize the value of k such that the probability that the eavesdropper succeeds in uncovering σ with 1-c probability happens just after k rounds. When c=99.99%, this means that we want to find the maximal value of k such that for all $j>k$, the probability that the eavesdropper finds σ at the jth round is more than 0.01%.

Assuming the regular graph F is connected, then we have the follow results:

Case 1: when regularity is large, $n-r \approx 0$:

$$\begin{aligned}
&\Pr[\text{ eavesdropper uncovers } \sigma \text{ after the } k\text{th round }] \\
&= 1 - (1 - (1/2)(S_F/n!))^k \\
&\leq 1 - (1 - (1/2)(n(r!)(r-1)^{n-r-1}/n!))^k \\
&\approx 1 - (1 - (1/2)(n!/n!))^k \\
&= 1 - (\tfrac{1}{2})^k
\end{aligned}$$

We see that the chance of an eavesdropper succeeding after just one round becomes very high (50%); more over, at this point the chance of an eavesdropper succeeding reduces to a series of coinflips. Furthermore, for the protocol to be secure within some confidence percentage c, we need to estimate the value of k such that Pr[eavesdropper uncovers σ after the kth round] $\geq 1 - c$. For the case of 99.99% confidence, we maximize the value of k for

$$1 - (1/2)^k \geq 1/10000, \quad (4)$$

and find that $k \leq 0.00014$. In other words, as soon as the protocol starts, it becomes impossible to maintain the 99.99% confidence threshold; by experimenting with small values of k we can see that the probability of an eavesdropper's success increases dramatically as the number of rounds increases:

Pr[eavesdropper uncovers σ after the first round] ≤ 0.5,
Pr[eavesdropper uncovers σ after the second round] ≤ 0.75,
Pr[eavesdropper uncovers σ after the third round] ≤ 0.875.

Case 2: when regularity r is small compared to n:

Pr[eavesdropper uncovers σ after the kth round]
$= 1 - (1 - (1/2)(S_F /n!))^k$
$\leq 1 - (1 - (1/2)(n(r!)(r\text{-}1)^{n\text{-}r\text{-}1}/n!))^k$

in this case, the chance of an eavesdropper succeeding after k rounds depends on the number of nodes in F and its regularity.

Again, we calculate the value of k for which the protocol achieves some level of confidence; let $c = 99.99\%$, then we want to maximize the value of k such that

$$1 - (1 - (1/2)(n(r!)(r\text{-}1)^{n\text{-}r\text{-}1}/n!))^k \geq 1/10000 \quad (5)$$

letting $c = 99.99\%$, we are able to create a bound on k,

$$k \leq \log(9999/10000) / \log(1 - (\tfrac{1}{2})(n(r!)(r\text{-}1)^{n\text{-}r\text{-}1}/n!)); \quad (6)$$

using Stirling's Approximation for factorials, and choosing small values of r such that $r \geq 3$, we generate the following graph

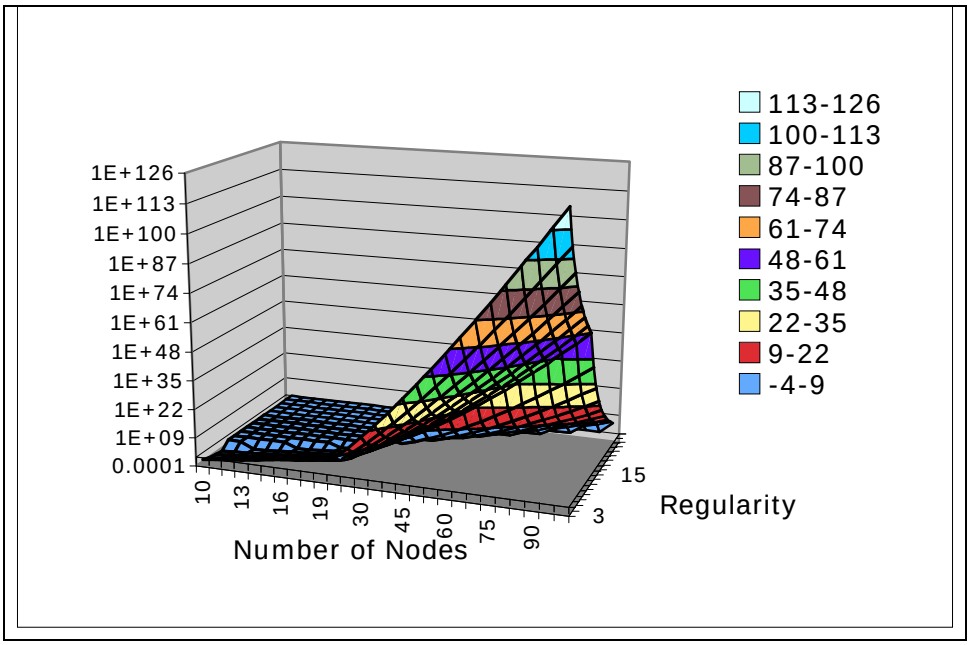

Figure 3: *Maximum number of rounds to maintain 99.99% confidence*

Even for a small value of *n*, the bound on *k* is quite good. In particular, for a graph with 14 nodes and regularity of 3, the number of rounds needed before an eavesdropper succeeds with more than 0.01% chance is 207. Furthermore, the graph illustrates that in general, as long as the value of *r* is limited to less than $\sqrt{n}$, then the value of *k* will also be large enough for use in the GI protocol.

When interpreting these results, keep in mind the inversion problem. If the graph can be inverted and solved more easily, then the confidence should be based on the difficulty in solving for the inverted graph, not the original. This problem is nullified by choosing r to be n/2.

3.1.4 Difficulty in generating good random regular graphs.

In the above analysis, we showed that when a connected graph is of small regularity, then its security characteristics are quite good. The characteristics of small-regularity graphs seem to be well understood [25, 27]. There also exist algorithms that generate random regular graphs under some constraints in non-exponential time [25, 26, 36]. However, we have not seen any universal random regular graph generation algorithm that meets all of our restrictions.

Further complicating the issue for generating good random regular graphs is the existence of efficient isomorphism finding programs, such as Nauty [28]. Nauty leverages any non-trivial automorphisms. It has been shown that Nauty is able to find isomorphisms for most graphs with less than 100 vertices in less than one second. For the graph isomorphism protocol to be secure, it is then necessary to generate random regular graphs with a large number of vertices while pruning the candidate graphs with large automorphism groups. This is a time consuming task, given that the known algorithms for generating random regular graphs spend most of their time in generating and eliminating unqualified graphs. Although the theoretical results show that some classes of graphs such as regular graphs may be good candidates for the GI protocol, there remains the obstacle that the current state of the art algorithms for random graph generation isn't efficient enough to warrant the use of GI as a basis for ZKP.

3.2 Factoring Requirements and Results

The implementation we chose was generic, not aimed for maximal performance. An identification scheme such as Guillou-Quisquater would be more efficient given that our implementation only ensures weak anonymity. Our goal was to show how the GI approach compared against other generic approaches, and more optimized approaches such as Guillou-Quisquater would have been an unfair comparison given the early state of the GI approach.

3.2.1 Requirements

3.2.1.1 Key Generation Requirements

The public keys are n and a. In general, large primes p and q are chosen so that n is difficult to factor using current methods [32]. We assume an n of at least 1024 bits. Since $a \equiv x^2 (\mathrm{mod}\, n)$, a also takes up at least 1024 bits. Thus, at least 2048 bits, or 256 bytes, of information must be made public before the rounds of zero knowledge proofs begin. As for private key storage, since $x < n$, then we assume x to take up at most 1024 bits.

Since there are a constant number of multiplications, the computation overhead is $O(c)$.

3.2.1.2 Protocol Initiation Requirements

The values of r and y are 1024 bits, since they are both taken modulo n. Computing r^2 is an $O(c)$ operation.

3.2.1.3 Answer Generation Requirements

In the third step, $z = r$ or $z \equiv xr (\mathrm{mod}\, n)$ is sent to V, so at most 1024 bits are sent over the communication lines.

3.2.1.4 Verification Requirements

No explicit storage is required at this step, and the computational overhead for answer verification is constant.

3.2.1.5 Performance Complexity

The security of this protocol relies on the primes chosen. That is, assuming 1024 bits is "safe enough" for today, then a total of 1024*4 + ε bits are sent over the communication lines over one round of zero-knowledge proof. Clearly, if a bigger prime is picked, say, of size *B*-bits, then the communications overhead will scale linearly, and over *k* rounds, is $k(4B + \varepsilon)$, or $O(B)$.

The computational overhead is $O(c)$.

3.2.2 Results

The curve here looks polynomial, as opposed to the O(c) predicted from the theoretical analysis of the protocol. This may be attributed by the computational overhead incurred by the physical limitation of the operational system. But note that even at 8192-bits the entire protocol completes in 7 milliseconds.

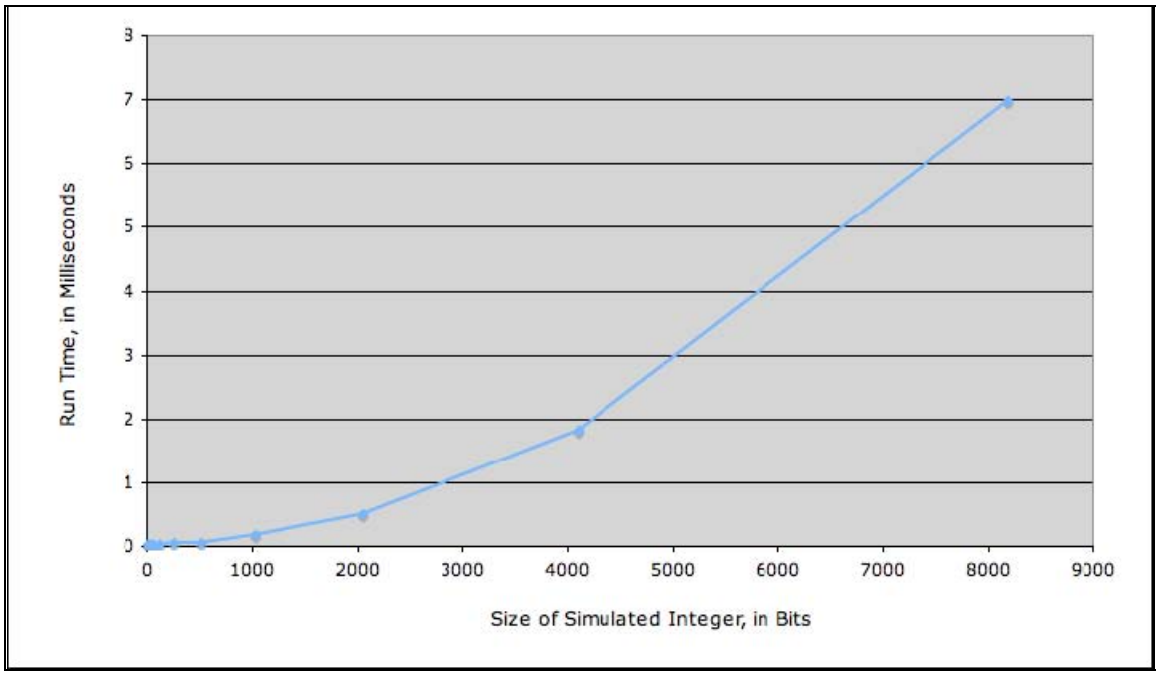

Figure 4: *Factoring Protocol Simulation*

Results related to security are presented in section 3.4 and compared with the other two techniques.

3.3 Discrete Logarithm Requirements and Results

As with factoring the implementation we chose was generic, not aimed for maximal performance. An identification scheme such as Schnorr would be more efficient given that our implementation only ensures weak anonymity. The goal was to show how the GI approach compared against other generic approaches, and more optimized approaches such as Schnorr would have been an unfair comparison given the early state of the GI approach.

3.3.1 Requirements

3.3.1.1 Key generation Requirements

The values α, β, and N are generated; N is normally chosen to be a large prime, typically 1024 bits and above. Thus the prover will publish 3*1024 bits of information.

For the private key, x has at least $\phi(N)$ possible values, where $\phi(N)$ is the number of values less than N that are relatively prime to N. When N is prime, $\phi(N) = N - 1$, so we can assume that x is about 1024 bits also.

3.3.1.2 Protocol Initiation Requirements

P selects r randomly from Z_N^*, so r is represented by 1024 bits. Note that Z_N^* are the non-zero integers less than N. γ is also represented by 1024 bits, since it is taken modulo N. So 1024 bits is the communications overhead.

The exponentiation of α^r takes $O(\log r)$ steps.

3.3.1.3 Answer Generation Requirements

Since y is taken modulo $\phi(N)$, it is represented by 1024 bits.

3.3.1.4 Verification Requirements

The exponentiation α^y takes $O(\log y)$ steps.

3.3.1.5 Performance Complexity

Like the Factorization Protocol, the communications overhead of the Discrete Logarithm protocol depends on the size of the prime. In the case of a 1024-bit prime, the total communication overhead incurred by the protocol is (5*1024 + ε) bits. Over k rounds, and for B-bit primes, the total communications overhead is $k(5B + \varepsilon)$ bits, or $O(B)$.

The total computation overhead in this protocol is $O(\log r + \log y) = O(\log ry)$.

3.3.2 Results

The graph shows an exponential growth in the running time, contrary to the expected logarithmic behavior from the theoretical analysis of the protocol. However, upon closer inspection, we do indeed see a logarithmic behavior up to about 2048 bits. This exponential growth after 2048-bits may be contributed by the fact that there isn't enough physical memory available for exponentiation of 2048-plus-bit numbers to a 2048-plus-bit number, even though in theory exponentiation takes logarithmic steps. In other words, the exponentiation procedure spends most of the time swapping memory in and out of the hard disk, resulting in the exponential curve that is shown in the graph. It is also worth noting that even with the exponential-like curve, the actual run time is still quite manageable; from the graph we see that at 4096-bits it still takes less than half a second to complete the entire protocol.

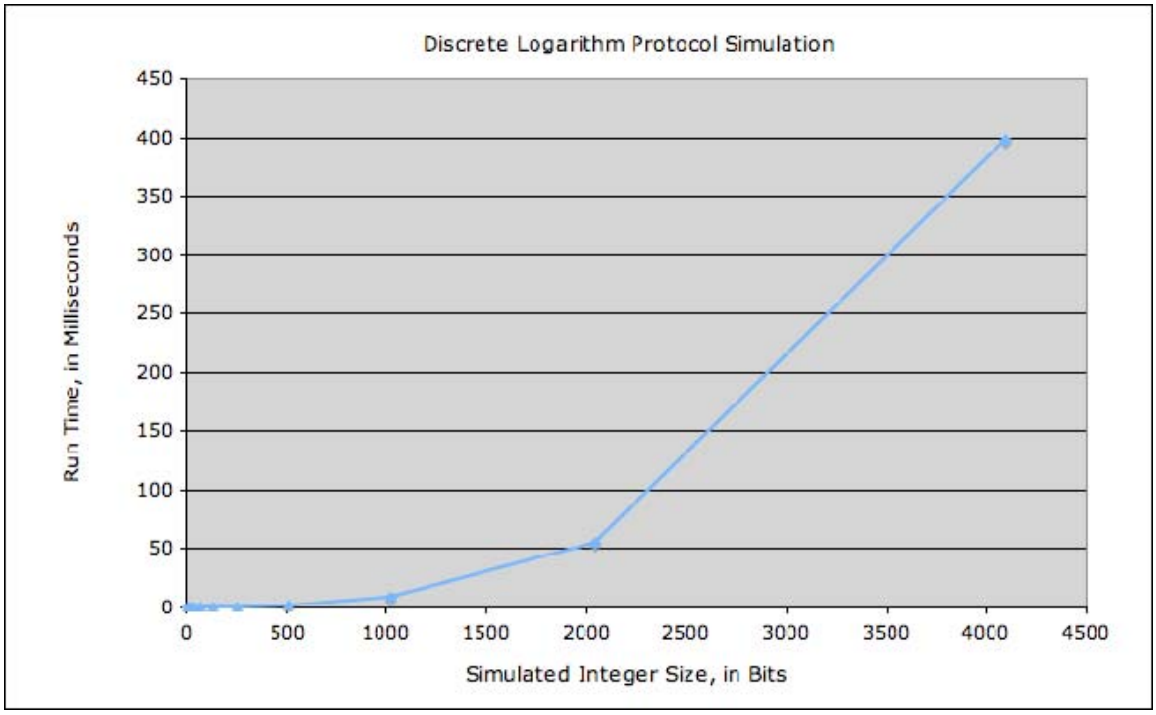

Figure 5: *Discrete Logarithm Protocol Simulation*

Results related to security are presented in section 3.4 and compared with the other two techniques.

3.4 Discussion

In the previous sections, we investigated the properties of three different applications of Zero-Knowledge Protocol: Graph Isomorphism, Factoring, and Discrete Logarithm. The performance complexities of the protocols are summarized in Table 1. We find that over *k* rounds, the Graph Isomorphism protocol has a smaller communication overhead (since relatively small graphs may be used) than Factoring and Discrete Logarithm. The GI may take significantly longer to generate good candidate graphs before the protocol can begin, however, because the efficient generation of random graphs with particular characteristics (such as regularity) is still an open research area [38]. On the other hand, generating large primes is a relatively fast process that takes polynomial time to complete [37].

Table 1: *Summary of Performance Complexities*

	Factoring Protocol	Discrete Logarithm Protocol	GI Protocol
Variable Definitions	*B*: size of modulus, in bits *c*: constant	*B*: size of prime, in bits *r*: size of integer, in bits *y*: size of integer, in bits	*r*: regularity of graph *n*: number of nodes in graph S_G: $\|Aut(G)\|$ S_F: $\|Aut(F)\|$
Communication Overhead	$O(B)$	$O(B)$	$O(rn)$
Computation Overhead	$O(c)$	$O(\log ry)$	$O(n^2)+O(1/S_G)+O(1/S_F)$

The computation overhead for Factoring and Discrete Logarithm protocols also fare better than GI. From our analyses, we see that the Factoring and Discrete Logarithm protocols require constant and logarithmic overhead, respectively, while GI requires at least polynomial computation overhead. Thus, given limited computing resources, the Factoring and Discrete Logarithm protocols scale far better than GI in terms of performance, and are better choices for the implementation of ZKP.

We also investigated the security properties of these protocols; the results are summarized in Table 2. The security properties of Factoring and Discrete Logarithm protocols are stronger than that of the GI protocol. Specifically, at the 99.99% confidence level, we see that the security of the GI protocol depends on the automorphism group of the candidate graph; when a "good" candidate graph is given, i.e. a graph with a small automorphism group, the GI protocol may be able to withstand the presence of an eavesdropper. However, given a "bad" candidate graph, the GI graph may fail to meet the 99.99% confidence threshold from the get-go. We also studied the application of regular graphs in ZKP, and found that the GI protocol is more secure when the graph has lower regularity. For example, when a regular graph of 14 nodes and regularity of 3 is used, the protocol is able to tolerate the presence of an eavesdropper for about 207 rounds.

Table 2: *Summary of Security Analysis.*

	Factoring Protocol	Discrete Logarithm Protocol	GI Protocol
Variable Definitions	N: 128-bit integer $\phi(N)$: Euler Phi function	N: 128-bit prime $\phi(N)$: Euler Phi function	F: random graph with 2^{128} nodes S_F: automorphism group of F, and we assume a relatively small automorphism group size, such that $S_F / n! \approx 10^{-10}$
1-Round Security	$1/\phi(N)$	$1/\phi(N)$	$(1/2)(S_F/n!)$
k-Round Security	$k/\phi(N)$	$k/\phi(N)$	$(1 - (1/2)(S_F/n!))^{k-1} \cdot (1/2)(S_F /n!)$
Max Number of Rounds Required to Maintain 99.99% Confidence Level	10^{35}	10^{35}	$2*10^7$

On the other hand, the security of Factoring and Discrete Logarithm protocols rely entirely on the size of $\phi(N)$, where N is the composite in the Factoring protocol and the prime in the Discrete Logarithm protocol. For either protocol, for example, if N is 1024 bits, then $\phi(N)$ must also be roughly 1024 bits. Furthermore, since the protocols do not reveal extra information after the primes are chosen, an eavesdropper must resort to brute force guessing of the prover's secret. The probability for the eavesdropper's success at one round is $1/\phi(N)$, an astonishingly low probability even when $\phi(N)$ is roughly 128-bits. If the eavesdropper attempts to exhaustively search through all possible numbers by guessing one number each round, then at the 99.99% protocol confidence, the number of rounds k required for the eavesdropper to succeed with more than 0.01% probability must satisfy

$$k/\phi(N) \geq 0.0001. \quad (7)$$

When $\phi(N)$ is 128-bits, k is roughly 10^{35}; the eavesdropper will not have any luck by guessing the prover's secret from the Factoring and the Discrete Logarithm protocols. This is significant because the prover typically exchanges zero-knowledge proofs with the verifier over a number of rounds, e.g. 20, that is much smaller than 10^{35}. Thus, for most ZKP exchanges, the Factoring and Discrete Logarithm protocols maintain the 99.99% confidence threshold.

In the general protocol, the probability that a malicious user successfully poses as a legitimate user is $(1/2)^k$, where k is again the number of rounds. We are interested in how many rounds it takes for the generic ZKP so that a cheater has less than 0.01% chance of passing all k rounds. This is found by solving for k in

$$(1/2)^k \leq 0.0001, \quad (8)$$

and that that $k \approx 13.28$. Therefore it takes at least 14 rounds for a cheater to have less than 0.01% chance of succeeding. Given this information, we calculate the number of bits exchanged for each protocol in 14 rounds:

Table 3: *Number of bits exchanged.*

	Factoring Protocol (128-bit prime used)	Discrete Logarithm Protocol (128-bit composite used)	GI Protocol (Graph with 128-bit nodes used)
Number of Bits exchanged in 14 rounds	14*4*128=7168 bits	14*5*128*14=8960 bits	Regular graph with r=n/2 edges = 64 bits 64*128 = 8192 bits

As we can see, when a sparse graph is used in the GI protocol, the bits exchanged is competitive with the other two protocols.

4. Discussion and Future Work

In this paper we touched on the issues of identification and anonymity in zkp protocols. In an identification scheme, each person carries a secret key *S* that is known only to that person; associated with each *S* is a public key *P* that is derived from *S* using a cryptographically-strong function, and the user authenticates himself to a server who holds the corresponding public key *P*. In a ZKP scheme, each user proves his identity to the server by answering questions about his secret key; the server verifies the answers by computing values based on *P* and other publicly available information. Note that in the ZKP scheme, the secret key *S* is never transmitted or revealed to the verifier. However, upon successful authentication of the user, 1 bit of information is revealed, namely, that particular person is a legitimate user to the system. While the protocol itself may not reveal additional information about the secret key (thus preventing any impersonation attacks), the identification of the user may lead to other actions by the eavesdroppers, such as physically following the user, which may endanger the success of missions.

An authentication scheme that maintains anonymity is desired over identification schemes. Two types of anonymity exist: weak anonymity and strong anonymity. Weak anonymity is essentially defined as hiding in masses. That is, assuming all legitimate users are given the same usernames and passwords it would be hard for the server and the observers to identify any particular user in that group. Thus, any legitimate user remains anonymous among its peer group of legitimate users. It is obvious that this protocol becomes harder to maintain its security as the user base expands. For example, if an individual's username and password are compromised, then the entire system is compromised. In this case, since the compromised user can't be revoked, there is no way to recover from this catastrophe. Therefore, any protocol with weak anonymity is clearly not scalable from a security maintainability perspective.

An authentication scheme guarantees strong anonymity when each user in the system is given a unique identification, yet is indistinguishable to an observer and the server. Very few ZKP algorithms currently in use support this, which partially explains why they are not in more wide-spread use. An interesting example of a protocol that has strong anonymity is described in [39]. The algorithm presented in this paper, however, is fairly inefficient compared to aggressively optimized identification schemes that are more commonly used in today's architectures. A good area of future work would be to explore methods for supporting characteristics such as strong anonymity and revocation, while still maintaining the efficient bandwidth characteristics of advanced identification protocols.

5. Conclusions

In this report, we investigated the use of the Zero Knowledge Protocol (ZKP) to authenticate and secure communication in a manner compatible with the requirements of an airborne network. We examined Factoring, Discrete Logarithm, and Graph Isomorphism protocols in detail. Theoretical analyses were conducted to study the applicability of these protocols; additionally, the simulations of the Factoring and Discrete Logarithm protocols were run to study the scalability performance of these protocols. We found that while the simulation results of Factoring and Discrete Logarithms did not match the theoretical results due to limitations of computing hardware, the protocols still finished in the ranges of milliseconds with the use of thousand-bit primes. Our conclusion based on these preliminary results is even without an optimized implementation Factoring and Discrete Logarithms protocols appear quite promising for application in the airborne networking domain.

We also derived theoretical results for the Graph Isomorphism protocol. Our analysis indicates that random graph selection is insufficient, but that a regular connected graph with a small automorphism group may hold the properties necessary to provide the desirable security properties. There are several drawbacks to this approach, however. First, generation of graphs with these properties is non-trivial. The current state of the art algorithms for regular graph generation is limited, and runs in polynomial time only for certain classes of regular graphs. To further complicate the problem, there are fast isomorphism detection programs such as Nauty that force the GI protocol to use regular graphs with large numbers of nodes for better protocol security. The efficient generation of regular graphs with large number of nodes and moderate regularity poses a significant challenge in and of itself. Therefore the GI protocol may not be feasible for real-world deployment due to the algorithmic challenges in generation of good candidate graphs.

A second drawback is that there is no evidence that the question-answer exchange cannot be captured, analyzed, and used to significantly improve the guessing process of a malicious user. This property is well established in the literature for Factoring and Discrete Logarithms, but until it can be shown for Graph Isomorphism we would be extremely hesitant to base a security protocol around it.

The Factoring and Discrete Logarithm protocols perform exceptionally well in terms of security and overhead requirements. These protocols are efficient in their computation overhead; the Discrete Logarithm protocol requires just a logarithm increase in computing overhead as the size of the prime increases; the Factoring protocol is even more efficient in that a constant amount of overhead is used no matter what size prime is used.

Our overall conclusion is that there is some indication that it may be possible that Graph Isomorphism could be made into an efficient algorithm. The number of bits sent per degree of security granted shows promise. There is quite a bit of work that would be necessary, however, to make this approach viable. First, new techniques in graph generation must be developed. Second, challenge-response exchanges must be developed that can be proven to be resistant to analysis of previously captured exchanges. If these feats are both accomplished (a non-trivial accomplishment) then a Graph Isomorphism based zero knowledge protocol may be competitive with, but not clearly superior to, more established techniques such as Factoring or Discrete Logarithm protocols. Even then, we believe there is more risk of future scientists discovering new heuristic mechanisms to compromise security characteristics than in the more established ZKP algorithms. This conclusion is based on the fact that Factoring and Discrete Logarithm problems have been heavily scrutinized by large numbers of scientists for hundreds of years and thus the difficulty of discovering heuristics that improve average case analysis is better established than the subset of Graph Isomorphic graphs we are considering.

Based on these findings, we consider Factoring and Discrete logarithm based zero-knowledge protocols to be the most promising of the algorithms investigated for potential application to airborne networking security protocols. We have implemented versions of both of these algorithms, and our preliminary results regarding speed and overhead reinforce these conclusions. We believe future work into the area expanding these approaches to efficiently support strong anonymity would be a valuable next step.

6. References

[1] O. Goldreich, *Zero-Knowledge twenty years after their invention*, 2002.

[2] O. Goldreich, S. Micali, and A. Wigderson, *Proofs that yield nothing but their validity*. Journal of the ACM, volume 38, issue 3, p. 690-728. July 1991.

[3] A. Sahai and S. Vadhan, *A Complete Problem for Statistical Zero Knowledge*, Journal of the ACM 50(2), 2003.

[4] O. Goldreich, R. Israel, and S. Goldwasser, *On the Limits of Non-Approximability of Lattice Problems*, Journal of Computer and System Sciences, p. 1-9, 1998.

[5] D. Micciancio, *The Shortest Vector in a Lattice is Hard to Approximate to within Some Constant*, Proc. 39th Symposium on Foundations of Computer Science, p. 92-98, 1998.

[6] C. Dwork, M. Naor, and A. Sahai, *Concurrent Zero-Knowledge*, 30th STOC, p.409-418, 1999.

[7] D. Micciancio and E. Petrank, *Efficient and Concurrent Zero-Knowledge from any public coin HVZK protocol*, Proceedings of *Advances in Cryptolog*, EUROCRYPT 2003, Poland, May 2003

[8] L. Babai and S. Moran, *Arthur-Merlin games: a randomized proof system and a hierarchy of complexity classes*, Journal of Computer and System Sciences, 36: p.254-276. 1988.

[9] D Pointcheval, G. Poupard, *A New NP-Complete Problem and Public-Key Identification*, Designs, Codes, and Cryptography Vol 28, Issue 1 (Jan 2003), p. 5 – 31.

[10] A. Shamir, *An efficient identification scheme based on permuted kernels* (extended abstract), Lecture Notes in CS Vol 435: Proc of the 9th Annual International Conf on Advances in cryptology, p. 606-609, July 1989, Santa Barbara, CA, Springer-Verlag, London, UK, 1989.

[11] J. Stern, *A New Identification Scheme Based on Syndrome Decoding*, Advances in Cryptology – proc. Of CRYPTO '93, LNCS Vol 773, p. 13 – 21, Santa Barbara, CA, 1994, Springer-Verlag.

[12] J. Stern, *Designing Identification Schemes with Keys of Short Size*, Advances in Cryptology – Proc. Of Crypto '94, LNCS Vol 839, p. 164 – 173, Santa Barbara, CA, 1994, Springer-Verlag.

[13] D. Pointcheval, *A New Identification Scheme Based on the Perceptrons Problem*, Advances in Cryptology- Proc. Of EUROCRYPT '95, LNCS vol 921, p. 319-328, Saint-Malo, France, 1995, Springer-Verlag.

[14] J. Stern, *An alternative to the Fiat-Shamir protocol*, Proc. Of Eurocrypt '89, LNCS Vol 434, p. 173 – 180, Springer-Verlag.

[15] G. Poupard, *A realistic security analysis of identification schemes based on combinatorial problems*, European Transactions on Telecommunications Vol 8, Issue 5, p. 471 – 480, 1997.

[16] T. Baritaud, et.al., On the Security of the Permuted Kernel Identification Scheme, Crypto '92, LNCS Vol 740, p. 305 – 311, 1992.

[17] E. Jaulmes and A. Joux, *Cryptanalysis of PKP: A New Approach*, LNCS Vol 1992, Proc of the 4th International Workshop on Practice and Theory in Public Key Cryptography, p. 165 – 172, 2001.

[18] L. Knudsen and W. Meier, Cryptanalysis of an identification scheme based on the permuted perceptron problem, Eurocrypt '99, LNCE vol 1592, p. 363 – 374, Springer-Verlag.

[19] D. Pointcheval, Neural networks and their cryptographic applications, Eurocode '94, INRIA (1994), p. 183 – 193.

[20] A. Fiat and A. Shamir, *How to Prove Yourself: Practical Solution to Identification and Signature Problems*, Crypto '86, LNCS Vol 263, p. 186 – 194, Springer-Verlag, 1987.

[21] "An Improved Protocol for Demonstrating Possession of Discrete Logarithms and Some Generalizations", D Chaum, J-H Evertse, J van de Graaf.

[22] "Proof Systems for General Statements about Discrete Logarithms", J. Camenisch and M. Stadler, Technical Report No. 260, March 1997, Dept. of Computer Science, ETH Zurich.

[23] "Efficient Signature Generation by Smart Cards", C.P. Schnorr, Journal of Cryptology, 4:161-174, 1991.

[24] "A Performance Comparison of Five Algorithms for Graph Isomorphism", P. Foggia, C. Sansone, and M. Vento, Proceedings of the 3rd IAPR TC-15 Workshop on Graph-based Representations in Pattern Recognition, 2001.

[25] “Models of Random Regular Graphs”, L. Shi, and N. C. Wormald, Surveys in Combinatorics, pp. 239-298, 1999.

[26] “Fast Generation of Regular Graphs and Construction of Cages”, M. Meringer, Journal of Graph Theory, 30:137-146, 1999.

[27] “Generating Random Regular Graphs”, J. H. Kim, and V. H. Vu, Combinatorica 26(6): 683-708, 2006.

[28] “Practical Graph Isomorphism”, B. D. McKay, 10th. Manitoba Conference on Numerical Mathematics and Computing (Winnipeg, 1980).

[29] “Upper Bounds on the Automorphism Group of a Graph”, I. Krasikov, A. Lev, and B. Thatte, Discrete Mathematics, 256:489-493, 2002.

[30] “The Asymptotic Number of Unlabelled Regular Graphs”, B. Bollobás, 26(2):201-206, 1982.

[31] “The Graph Isomorphism Disease”, R. Read and D. Corneil, Journal of Graph Theory, 1:339-363, 1977.

[32] “On Polynomial Selection for the General Number Sieve”, T. Kleinjung, Mathematics of Computation, 75:2037-2047, 2006.

[33] “Introduction to Algorithms,” T. Cormen, C. Leiserson, R. Rivest, and C. Stein.

[34] E. Brickell, J. Camenisch, and L. Chen, *Direct Anonymous Attestation*. http://eprint.iacr.org/2004/205/

[35] E. Brickell, J. Camenisch, and L. Chen, *Direct Anonymous Attestation*, In CCS ’04: Proceedings of the 11th ACM conference on Computer and communications security. ACM Press, New York, United States of America, pp. 132–145.

[36] A. Steger and N.C. Wormald, *Generating Random Regular Graphs Quickly*, 1999.

[37] W. Diffie and M. E. Hellman, *New Directions in Cryptography*. IEEE Trans. Inform. Theory, 22(6):644-654, 1976.

[38] “Graph Theory,” A. Bondy and U. S. R. Murty.

[39] R. Impagliazzo and S. Miner More, *Anonymous Credentials with Biometrically-Enforced Non-Transferability*, 2003 ACM Workshop on Privacy in the Electronic Society.

[40] J. Staddon, S. Miner, M. Franklin, D. Balfanz, M. Malkin and D. Dean, *Self-Healing Key Distribution with Revocation*, IEEE Symposium on Security and Privacy 2002.

[41] S. Miner More, M. Malkin, J. Staddon and D. Balfanz, *Sliding-Window Self-Healing Key Distribution*, 2003 ACM Workshop on Survivable and Self-Regenerative Systems.

[42] M. Bellare and A. Palacio, *GQ and Schnorr Identification Schemes: Proofs of Security against Impersonation under Active and Concurrent Attacks*, Advances in Cryptology - Crypto 2002 Proceedings, Lecture Notes in Computer Science Vol. 2442, M. Yung ed, Springer-Verlag, 2002.

7. List of Symbols, Abbreviations, and Acronyms

CA: Certificate Authority
CLE: Constained Linear Equations
CVP: Closest Vector Problem
DAA: Direct Anonymous Attestation
FAAN: First Time Authentication for Airborne Networks
GI: Graph Isomorphism
ID: Identification
MANET: Mobile Adhoc Network
$\phi(N)$: Euler Phi function
PKP: Permuted Kernels Problem
PP: Perceptron Problem
PPP: Permuted Perceptron Problem
SD: Syndrome Decoding
SVP: Shortest Vector Problem
TCG: Trusted Computing Group
TPM: Trusted Platform Module
ZKP: Zero Knowledge Protocol
Z_N^*: non-zero integers modulo N